

APPENDIX 1

```

void sw ( )
(
5
#define iw = 12;                                /* instruction
                                                width */

#define mw = 3:                                 /* memory width */
#define CONST = 0                                /* push constant */
#define LOAD = 1                                 /* push variable */
#define GLOBAL = 2                               /* push address */
#define PUTCHAR, = 15 /* put a character along the
                        standard output channel */

10
#define GETCHAR = 16 /* get a character from the
                        standard input channel */

15
...
rom program []
20
#include "prog.o" ): ram stack[1<mw] with dualport = 1 ];
ram memory[1<mw] unsigned iw PC, ir, tos;
unsigned mw sp;

25
do par it = program[pc]: PC = PC + 1;
tos = stack[sp-1];                                /* save top of
                                                stack to avoid
                                                two ram accesses
                                                in one cycle
                                                */

30
switch (ir)
case
CONST par
35
    stack[sp] = program[pc];
    sp = sp+1;
    PC = PC+1;
    ]
    break;
case LOAD
40
    stack[sp-1] = memory[tos<-mw];
    break;
case STOP break; default :                         /* unknown opcode */
while (1) delay;

45
] while (ir != STOP);

```

]

Register transfer level description of simple processor

5

APPENDIX 2

```
void main() { char hswchan;
char unsigned 8 port:
5
    par {
        parallel_>,port(port);
        SyncGen():
10
            initialiseRam(port);
            par {
                display(hswchan): sw(hswchan);
                y 1 }
            }
15
    RTL description of main
```

APPENDIX 3**CALCULATION PROCESS**

```

5  /*
 * Channel communicating object positions
 */ chap unsigned 17 position;

10 /*
 * Channel communicating segment information
 */
chanout unsigned 9 segment;

15 /*
 * Channel communicating button information
 */
chanin unsigned 2 buttons;

20 /*
 * Overall par
 */ par

25 /*
 * Mass motion
 */

30 /*
 * Positions of each mass, 9+8 fixed point
 */
unsigned 17 p0, p1, p2, p3, p4, p5, p6, p7;
/*
 * Velocity of each mass, 9+8 fixed point
 */
int 17 v1, v2, v3, v4, v5, v6, v7;

35 /*
 * Accelerations of each mass, 9+8 fixed point
 */
int 17 a1, a2, a3, a4, a5, a6, a7;

40 /*
 * Sutton status
 */
unsigned 2 button status;

45 /*
 * Initial setup of positions
 */

```

```

p0 = 65536;
p1 = 65536;
p2 = 65536;
p3 = 65536;
5   p4 = 65536;
p5 = 65536;
p6 = 65536
p7 = 65536

10
/*
 * Forever
 */
15 while (1)
{
/*
 * Send successive positions down position channel
 */
20   send(position, p0);
send(position, p1);
send(position, p1);
send(position, p2);
send(position, p2);
25   send(position, p3);
send(position, p3);
send(position, p4);
send(position, p4);
send(position, p5);
send(position, p5);
30   send(position, p6);
send(position, p6);
send(position, p6);
send(position, p7);

35
/*
 * Update positions according to velocities
 */
40   p1 +_ (unsigned 17)vl;
p2 +_ (unsigned 17)v2;
p3 +_ (unsigned 17)v3;
p4 +_ (unsigned 17)v4;
p5 +_ (unsigned 17)v5;
p6 +_ (unsigned 17)v6;
p7 +_ (unsigned 17)v7;
45
/*

```

```

* Update velocities according to accelerations
*/
5    v1 += a1 - (v1 » 6);
    v2 += a2 - (v2 » 6);
    v3 += a3 - (v3 » 6);
    v4 += a4 - (v4 » 6);
    v5 += a5 - (v5' » 6);
    v6 += a6 - (v6 » 6);
    v7 += a7 - (v7 » 6);
10   /*
* Set accelerations according to relative positions
*/
15   a1 = (int 17)((p2 » 8) - (p1 » 8)) + ((p0 » 8) - (p1 » 8));
    a2 = (int 17)((p3 » 8) - (p2 » 8)) + ((p1 » 8) - (p2 » 8));
    a3 = (int 17)!!(p4 » 8) - (p3 » 8)) + ((p2 » 8) - !p3 » 8));
    a4 = (int 17)((p5 » 8) - (p4 » 8)) + ((p3 » 8) - (p4 » e>>));
    a5 = (int 17)((!p6 » 8) - (p5 » 8)) + ((p4 » 8) - (p5 » 8));
    a6 = (int 17)((p7 » 8) - (p6 » 8)) + ((p5 » 8) - (p6 » e >));
    a7 = (int 17)((p6 » 8) - (p7 » 8));
20   /*
* Get button information
*/
25   receive(buttons, button status);

/*
* Fix top point according to buttons
*/
30   if (button status & 1)
35   else
        if (button status & 2)
        (
            p0 = 65536 - 16384;
        )
        else
            p0 = 65536 + 16384;
40   }
45   /*
* nine drawing
*/
(
    /*

```


DISPLAY PROCESS

```

5      /* standard includes */
         #include "hammond.h"
         #include "syncgen.h"
         #include "stdlib.h"
10     #include "parallel.h"

         /*
* Segment information channel */ chap segment;

15     /*
* Button information channel */
chan buttons:

         /
20     * Include dash generated stuff */
         #include "handelc.h"

         /*
* Main program */
25     void main() (
         /
         * Scan positions
         */ unsigned sx, sy;

30     /*
* Vdeo output register
*/
         unsigned 1 video;

35     /*
* Video output bus
*/
         interface bus out() video out(Visible(sx, sy) ?
40     (video ? (unsigned 12)Oxffff : 0) 0) with video spec;

         #ifndef SIMULATE
         /*
* Left button input bus
*/
45     interface bus in (unsigned 1) button_left()

```

```

with button white spec;

5   /*
* Right button input bus
* */
   interface bus in(unsigned 1) button right()
      with button_black spec;
   #endif

10  /*
*
   Overall par
*/ par {
/*
15   * VGA sync generator
*/
   SyncGen(sx, sy, hsync pin, vsync pin);
/*
*
20   * Dash generated hardware
*/
   hardware();
/*
25   * Run-length decoder
*/
{
/*
   * Segment start and end positions
*/
30   unsigned start, end;
/*
   * Forever
*/
35   while (1)
{
   while (sy != 448)
/*
      * Read segment information
*/
40   segment ? start;
   segment ? end;
/*
      * Get in the right order
*/
45   if (start > end)
{

```

```

    par
    {

5    end = start;
    start = end;
    )

10   /* Make at least 1 pixel visible
    */
    if (start == end)
        end++;

15   /*
        * Wait
    */
    while (sx != 0)
        delay;
    /*
        * Draw a scanline worth
        */
    while (sx != 512)
        if ((sx < 9) >= start && (sx < 9) < end)
25
        video = 1;
        else
        video = 0;

30
        )
    /*
        * Communicate button status
    */
#ifndef SIMULATE
35    buttons ! 1;
#else
        buttons ! button left.in @ button right.in;
#endif
        /*
        * Wait
        */
40    while (sy != 0)
        delay;
    )
45

```